ETH zürich

Soft Robotics Laboratory

Semester Project

Development of a Model Predictive Controller for FALCON: Fixed-wing Aerial Lifting and Carrying of Objects inspired by Nature

Aurel X. Appius

January 2023

Supervision

Yasunori Toshimitsu
Prof. Dr. Robert K. Katzschmann

Abstract

Aerial grasping of objects is a problem, where nature is still miles ahead of robotics. When comparing an eagle grasping a fish out of the water with a modern grasping robot, the differences in agility, speed, and precision are clearly noticeable. The pick-up speed, the grasping success rate, and the ability to cope with disturbances caused by the environment are some examples of animal superiority over state-of-the-art robots. To bring robots one step closer to their natural counterpart, we want to autonomously grasp an object using a fixed-wing aircraft. To achieve this next step, we want to use a model predictive controller on a lightweight model plane equipped with a soft robotic gripper. A coordinated movement between the plane and the gripper should decrease the relative speed of the gripper to the object, resulting in a high gripping success rate. Such a movement is achieved using the novel fixed-wing control approach presented in this work. By using a fixed-wing aircraft, we can perform the pick-up more energy efficiently than quadcopters and therefore stay in the air for longer periods. After analyzing the dynamics of the fixed-wing aircraft, an MPC and an offline trajectory approach were used on a custom simulation. While the MPC approach did not yield results due to the convergence speed of the optimization, the offline trajectory controller achieved a 60% grasp success rate in simulation. Furthermore, a real-world system was built and deployed to fly simple autonomous trajectories. This work paved a first path in the direction of successful autonomous fixed-wing aerial grasping, by providing meaningful methodology, simulation results as well as a prototype to continue the research.

Contents

1 Introduction

1.1 Motivation

Roboticists throughout history have commonly looked to nature when they needed the inspiration to design new robots. Through more than a billion years of evolution, animals became skilled and efficient in performing difficult dynamic maneuvers. An impressive feat is the grasping ability of predatory birds, who can pick up objects from the ground in a dynamic swooping motion. Fast maneuvers like these require both high precision and an agile body. Fig. 1 shows a sequence of a bald eagle, that performs a grasping maneuver. When looking at the sequence, we observe that the eagle approaches the fish with his claws in the front. During the pick-up, his body moves over the fish, while the claws perform a backward motion to compensate for the forward speed. When the eagle finishes the grasping maneuver, the claws are in the back. Another interesting observation is the wing deformation of the eagle during the pick-up. During the approach, the eagle is sailing towards the target with almost horizontal wings. He then lifts them up, increasing the drag and therefore slowing him down. After the fish has been grasped, he flaps his wings back into the horizontal position giving him acceleration. Summed up, he decelerates until the object is grasped and then starts accelerating again. This is a desirable behavior that we would like to see in our control approach, by choosing the right cost function.

Figure 1: An Eagle grasping salmon from the end of a boat. The frames were taken from a video found in New York Post (2017).

1.2 Related Work

Various researchers have already tried to create robots that grasp like predatory birds. The most common approach is the combination of a gripper with a quadcopter. Gawel et al. (2017) used a magnetic gripper to pick up ferrous objects while moving. Fiaz et al. (2018) combined a permanent magnetic gripper with a dropping mechanism to let go of objects on demand. Roderick et al. (2021) achieved biomimetic perching of branches by using a combination of an arm and a gripper. Generally speaking, there have been many works on quadcopter-based aerial grasping and perching.

Hingston et al. (2020) designed a net- and a slider-based gripper for aerial grasping with quadcopters. Another gripper mechanism for quadcopters by Zhang et al. (2019) was used for perching onto power lines to recharge the battery. Another interesting design by McLaren et al. (2019) uses a passive gripper to achieve very fast grasping speeds of only 96 milliseconds.

All the mentioned robots that can perform grasps and perches are quadcopters. They can hover and can temporarily halt during a grasping maneuver. Some quadcopters can grasp without stopping over the object (Appius et al., 2022), (Fishman et al., 2021), (Thomas et al., 2014). This allows a fast pickup but requires a lot of energy. Fixed-wing aircraft outperform quadcopters drastically when it comes to specific energy consumption as well as energy per distance (Karydis & Kumar, 2017). An eagle can both grasp an object dynamically and then fly long distances without needing a recharge every 10 minutes.

Another issue with quadcopter grasping is the downwash of the rotors (Zhu et al., 2022). The high winds caused by the propellers can blow away the object that the quadcopter wanted to grasp making the task nearly impossible. This effect becomes particularly noticeable when picking up lightweight objects.

Stewart et al. (2022) used a fixed-wing aircraft in combination with a passive gripper to grasp objects dynamically. They achieved grasping speeds of 8 m/s, but due to the passive gripper, the robot can't drop the object and is therefore limited. The rigid gripper is also limited in grasping fragile or delicate objects, due to its rigid nature.

Moreover, the system proposed by Stewart et al. (2022) is not fully autonomous and relies on a human pilot to fly. Due to the influence of the human pilot and the lack of comparability between grasps, they only performed a single aerial grasp.

A review paper by Meng et al. (2021) summarized the current state of aerial grasping robots. According to Meng et al. (2021) and my knowledge, a fully autonomous pick-up of objects using a fixed-wing aircraft has yet to be achieved.

Compared to drones, there is very little research that has been done for fixed-wing grasping, for which several reasons could be considered. The recent rise in popularity of quadcopters has probably pushed fixed-wing aircraft out of the focus. Experiments with planes need a lot of space and have to be conducted mostly outdoors, which brings in more disturbances making the task harder.

2 Goals and Deliverables

I propose **FALCON**: **F**ixed-wing **A**erial **L**ifting and **C**arrying of **O**bjects inspired by **N**ature. A fixed-wing aerial robot, that is equipped with a soft robotic gripper. By using a model predictive controller, the robot will be able to grasp static objects off the ground autonomously. An even more bio-inspired robot would use a flapping-wing architecture like the one proposed by Folkertsma et al. (2017). Since fixed-wing aerial grasping is still an open challenge, the even more difficult problem of flapping-wing grasping should be taken in a further step.

The robot should be able to achieve the following goals:

Grasping Using a soft robotic gripper, FALCON should be able to grasp objects of different sizes and geometries consistently. The objects should be approximately cylindrical with an approximate length of 20cm, a diameter of 8cm, and a weight of 200g. For such objects, we want to achieve at least 7 successful grasps in 10 attempts, achieving a 70 % success rate.

Autonomy FALCON should perform the whole flight and grasping maneuver fully autonomously.

3 Methodology

To achieve the goals mentioned in Section 2, a well-thought-out methodology is of great importance. The hardest challenge will be the grasp of the object, while the plane has high horizontal speeds. The most intuitive solution to reduce the pick-up speed would be to slow down the plane. Due to the aerodynamical lift required to fly, this can only be done to a certain level. Afterwards, more sophisticated methods are required to counteract the velocity. An example of such a method would be a coordinated movement of an arm attached to the plane. When swinging the arm back, the speed of the tip of the arm is lower than the speed of the plane. The following subsections describe this method with the required mathematical modeling. Another addressed challenge is the design of the system architecture that is required to deploy this grasp method.

3.1 Grasp Plan

To account for the fast speeds during the grasping maneuver, the gripper is placed on an arm. The idea is that the arm can move back while the aircraft passes over the object which results in a lower relative speed between the object and the gripper. This is exactly the movement, that we could observe for the eagles' pick-up of a fish in Fig. 1. This kind of motion gives the gripper more time to close and a successful pickup is likelier. An illustration of the principle can be seen in Fig. 2. A similar principle was used by Roderick et al. (2021) but for perching instead of grasping. They approached tree branches with the gripper in the front. After the first contact, the arm moved back to slow down the quadcopter which enabled successful perching. Lab (2013) used the same principle on a quadcopter to grasp at high speeds.
In the first approach, a single degree-of-freedom (DOF) arm is used, that can swing back and forth along the plane's x-axis. If the grasping precision of such an arm is too low during the aerial grasping experiments, more complicated arms with a second joint or axis are considered. But ideally, the uncertainties should be compensated by the soft gripper, which passively deforms according to the load.

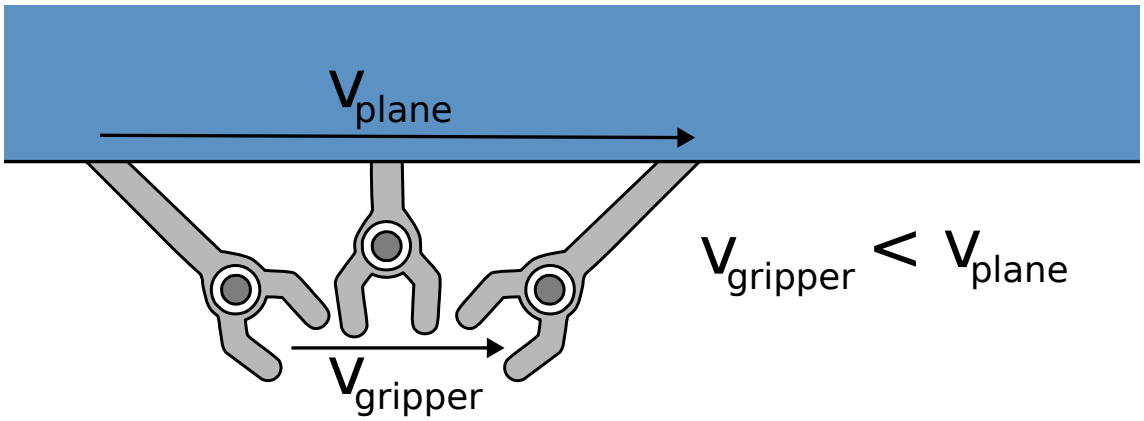

Figure 2: If the arm of the gripper rotates during the grasping sequence, the relative speed of the gripper is lower than the speed of the plane.

3.2 Flight Plan

The fixed-wing aircraft cannot hover like a quadcopter, so there needs to be a defined trajectory before and after the grasp. The simplest solution is to fly circuits over the target, which can be seen as the blue path in Fig. 3. Once the grasping sequence is initiated, the aircraft will change its trajectory and will perform a swooping motion which can be seen as the red path in Fig. 3. This is a pretty common practice for manned airplanes when approaching an airport and trying to land. If the landing maneuver fails, the plane ascends again and continues to fly a circuit above the runway to then retry the landing. Note that predatory birds also use this tactic when hunting for rodents on the ground. The main challenge in this section will be the implementation of this trajectory in the PX4 flight controller. Also, the landing and starting sequence has to be implemented to get the plane into the circuit motion and land it safely after the grasping is over.

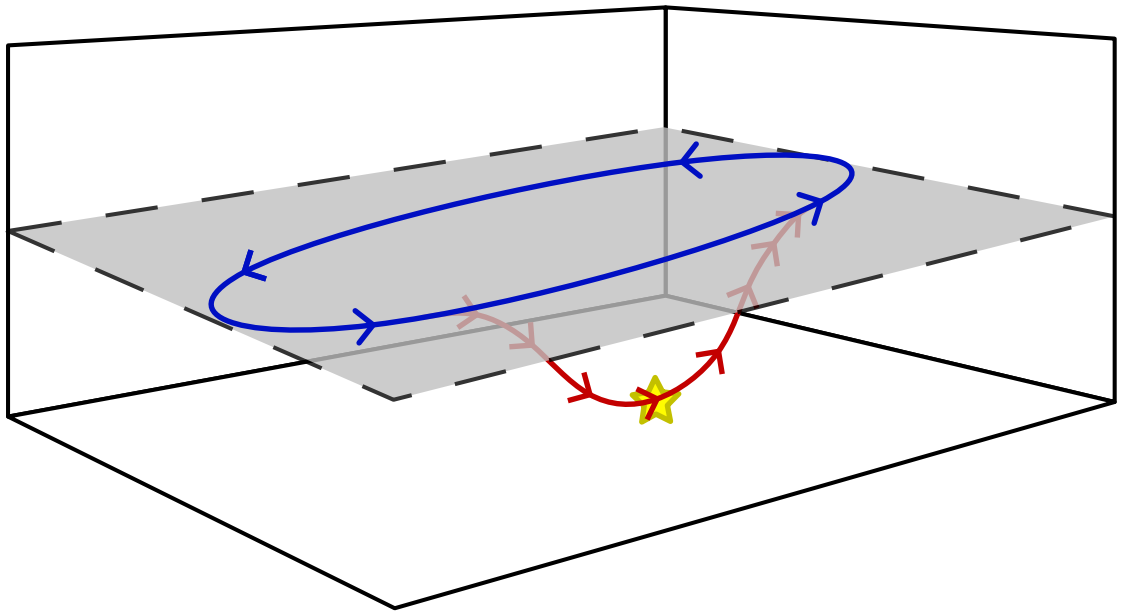

Figure 3: Planned flight path for the fixed-wing aircraft when grasping an object.

3.3 Modeling and Control

3.3.1 Gripper and Arm Dynamics

The gripper is modeled as a point mass on an arm that is attached to the aircraft. The input u_{arm} controls the angle β of the gripper's arm by applying a torque at the pivot of the arm. For the controller described in Section 3.3.6, the speed of the gripper is of great importance. By using the velocity transfer formula seen in Equation (1), we can derive the grippers speed as a function of the plane's speed v_{plane} and the angular velocity $\dot{\beta}$.

$$v_{grip} = v_{plane} + \dot{\beta} \times r_{plane-grip} \tag{1}$$

Finally, we can derive the state space representation of the gripper dynamics, which can be seen in Equation (2). Because the grasped objects are relatively light in comparison to the model airplane, we will consider the additional mass after the pick-up as a disturbance.

$$\frac{d}{dt}\begin{bmatrix} \beta \\ \dot{\beta} \end{bmatrix} = \begin{bmatrix} \dot{\beta} \\ \frac{\cos(\beta)\cdot g}{m_{gripper}\cdot l_{arm}} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{m_{gripper}\cdot l_{arm}^2} \end{bmatrix} \cdot u_{arm} \tag{2}$$

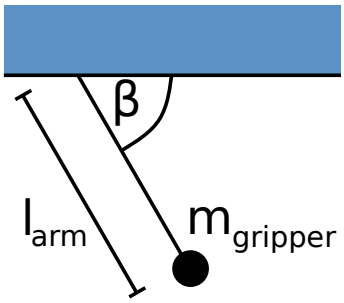

Figure 4: This sketch shows the proposed physical model for the gripper and arm.

3.3.2 Plane Dynamics

To properly control the motion of the aircraft, a mathematical model of the plane's dynamics has to be derived. For this let us first define the coordinates of the plane. We do this by having an inertial frame and a body frame that is fixed to the aircraft. We denote the position of the aircraft with the vector $\vec{X}_{plane}$. As shown by Noth et al. (2006), the equations of motion can be derived by using the Lagrange-Euler approach. The Lagrangian is defined as seen in Equation (3) where T denotes the kinetic and V denotes the potential energy.

$$\mathcal{L} = T - V \tag{3}$$

The equations of motion are found by inserting the Lagrangian into the differential equation seen in Equation (4), where q denotes the generalized coordinates and Γ denotes the non-conservative forces. In our case, the non-conservative forces come from the aerodynamic interactions of the plane with the environment.

$$\Gamma_i = \frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{q}_i}\right) - \frac{\partial \mathcal{L}}{\partial q_i} \tag{4}$$

When solving and simplifying Equation (4) for all state space coordinates q_i, which are defined in Table 1, the state space dynamics of the plane are found. The final equations can be seen in Equation (5).

Variable	Description	Variable	Description
ϕ	Roll Angle	x	x-position
θ	Pitch Angle	y	y-position
ψ	Yaw Angle	z	z-position
$\dot{\phi}$	Roll Rate	$\dot{x}$	x-velocity
$\dot{\theta}$	Pitch Rate	$\dot{y}$	y-velocity
$\dot{\psi}$	Yaw Rate	$\dot{z}$	z-velocity

Table 1: State space variables of the plane's dynamics

In the equations of motion seen in Equation (5), gravity and aerodynamic forces are summed up as a force and a torque on the center of mass of the plane. The force F_{tot} as well as the torque M_{tot} are described in greater detail in the section Section 3.3.4.

$$\begin{aligned}
\ddot{x} &= \frac{F_{tot,x}}{m} \\
\ddot{y} &= \frac{F_{tot}, y}{m} \\
\ddot{z} &= \frac{F_{tot,z}}{m} \\
\ddot{\phi} &= \frac{I_{pitch} - I_{yaw}}{I_{roll}} \dot{\psi}\dot{\alpha} + \frac{M_{tot,x}}{I_{roll}} \\
\ddot{\alpha} &= \frac{I_{yaw} - I_{roll}}{I_{pitch}} \dot{\psi}\dot{\phi} + \frac{M_{tot,y}}{I_{pitch}} \\
\ddot{\psi} &= \frac{I_{roll} - I_{pitch}}{I_{yaw}} \dot{\alpha}\dot{\phi} + \frac{M_{tot,z}}{I_{yaw}}
\end{aligned} \tag{5}$$

3.3.3 Plane Actuators

A fixed-wing aircraft has four main actuators, that it can use to control its flight:

The Propeller is used to accelerate the plane in the x-direction of the body frame.

The Ailerons are used to control the aircraft's roll.

The Rudder is used to control the yaw of the plane.

The Elevator can change the pitch of the aircraft.

Fig. 5 shows the four actuators, the body coordinate frame, and the names of the rotations around each axis. Of course, the actuation of a single actuator does not only result in the desired change of attitude but also has some adverse effects. For example, a generally known effect is that the use of the ailerons results not only in a change of the roll angle but also in a change in the yaw. Skilled pilots as well as control systems automatically compensate for that while flying.

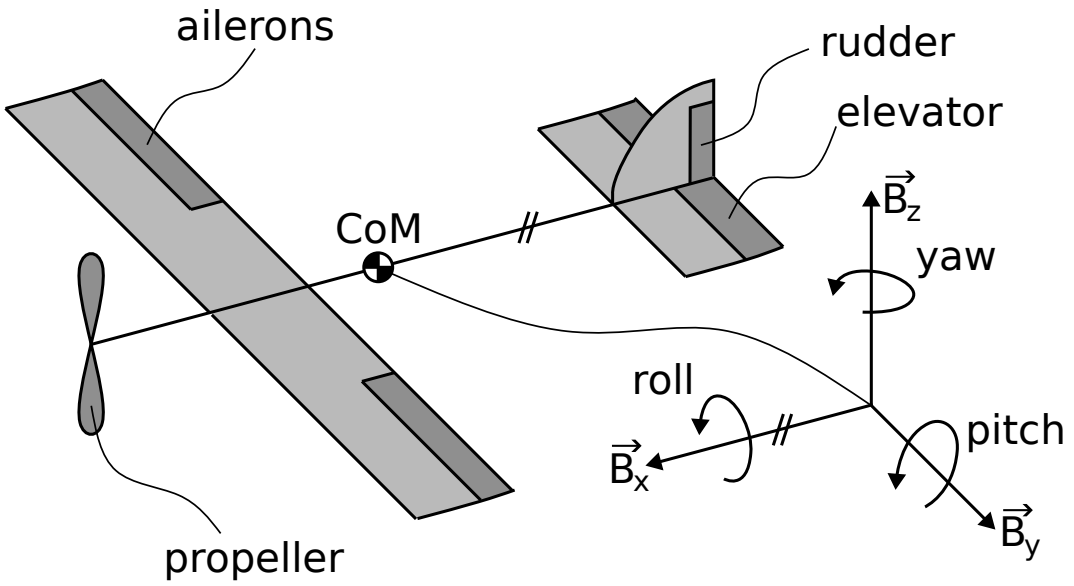

Figure 5: This sketch shows the simplified plane with all the parts that are important when mathematically modeling the plane as a dynamical system.

3.3.4 Fixed-Wing Aerodynamics

To find a full mathematical representation of the system, the dynamic response of the control inputs needs to be thoroughly studied. All controllable inputs can be seen in Table 2. Only the inputs u_1 to u_5 are used to control the plane's flight path.

Input Variable	Description	Unit
u_1	Motor Thrust	Newton
u_2	Left Aileron	Radian
u_3	Right Aileron	Radian
u_4	Elevator	Radian
u_5	Rudder	Radian
u_{arm}	Gripper Arm Angle	Radian
u_{grip}	Gripper Open/Close	[-]

Table 2: Inputs of the system.

To control the aircraft, we need a model that is as simple as possible, but rich enough to describe the dynamics of the plane. I propose a model that can be seen in Fig. 6. The lift and drag of the airplane are determined experimentally. They are a function of the angle of attack of the aircraft as well as the speed at which it travels. Additionally, the control surfaces have an impact on the flight behavior, which is modeled by doing a conservation of momentum. The following list discusses the relationship between the control inputs and the forces acting on the plane. The idea behind the modeling is to model the aircraft as an airfoil and the control inputs as perturbations. All forces seen in Fig. 6 are discussed in the following.

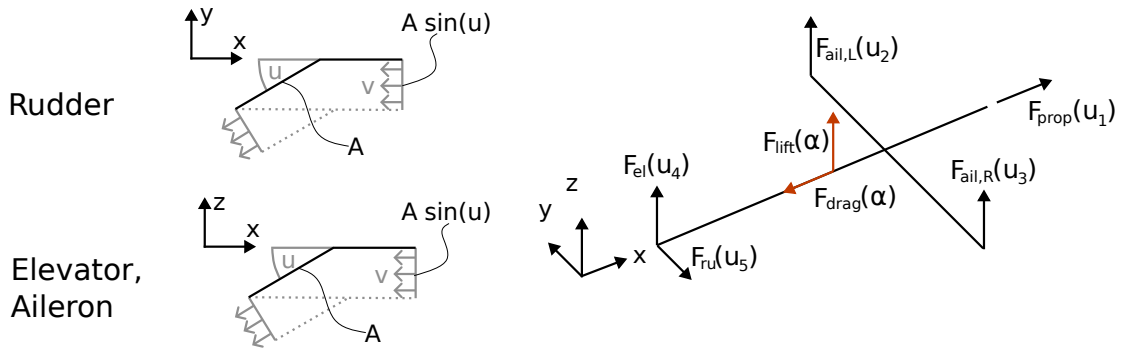

Figure 6: A simple model of an aircraft and figures clarifying the derivations in equations 6, 7 and 8.

Rudder The rudder is displacing air at the end of the airplane. We can model the force that this causes on the tail of the airplane by just comparing the incoming and outgoing momentum of the air affected by the deflection of the tail. In Equation (6), A_5 refers to the area of the rudder and v to the speed of the plane. If we assume that u_5 is a small angle, we can use the first-order Taylor approximation of the sine and cosine to further simplify the expression. This gives us a good estimate of the force caused by the deflection of the rudder.

$$\begin{aligned}\begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix} &= \underbrace{A_5\rho\sin(u_5)v^2\begin{bmatrix} -\cos(u_5) \\ \sin(u_5) \\ 0 \end{bmatrix}}_{\dot{p}_{out}} - \underbrace{A_5\rho\sin(u_5)v^2\begin{bmatrix} -1 \\ 0 \\ 0 \end{bmatrix}}_{\dot{p}_{in}} \\ &= A_5\rho\sin(u_5)v^2\begin{bmatrix} 1-cos(u_5) \\ sin(u_5) \\ 0 \end{bmatrix} \approx A_5\rho u_5 v^2\begin{bmatrix} 0 \\ u_5 \\ 0 \end{bmatrix} = A_5\rho u_5^2 v^2\begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}\end{aligned} \tag{6}$$

Elevator We can do the same analysis that we did for the rudder with the elevator, which is just a rudder that was rotated by 90 degrees. We therefore can approximate the force caused by the elevator as seen in Equation (7).

$$\begin{aligned}\begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix} &= \underbrace{A_4\rho\sin(u_4)v^2\begin{bmatrix} -\cos(u_4) \\ 0 \\ \sin(u_4) \end{bmatrix}}_{\dot{p}_{out}} - \underbrace{A_4\rho\sin(u_4)v^2\begin{bmatrix} -1 \\ 0 \\ 0 \end{bmatrix}}_{\dot{p}_{in}} \\ &= A_4\rho\sin(u_4)v^2\begin{bmatrix} cos(u_4)-1 \\ 0 \\ sin(u_4) \end{bmatrix} \approx A_4\rho u_4 v^2\begin{bmatrix} 0 \\ 0 \\ u_4 \end{bmatrix} = A_4\rho u_4^2 v^2\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}\end{aligned} \tag{7}$$

Ailerons The ailerons are modeled the same as the elevator. The forces caused by the left and the right aileron can be seen in Equation (8).

$$\begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix} \approx A_2\rho u_2^2 v^2\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \qquad \begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix} \approx= A_3\rho u_3^2 v^2\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{8}$$

Propeller The relation between the control input u_1 and the thrust of the propeller is found experimentally. By attaching the plane to a force gauge with a tether and then actuating the propeller, we can observe force measurements for different thrust inputs. A quadratic function is used to fit a curve into the data points. The found relation can be seen in Fig. 7 and in Equation (9). I conducted these experiments with the real plane. For the simulation, I could see the propeller constants determining the thrust in the source code so I didn't need to perform an experiment there.

$$F_{prop}(u_1) = 14.456 \cdot u_1^2 \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \tag{9}$$

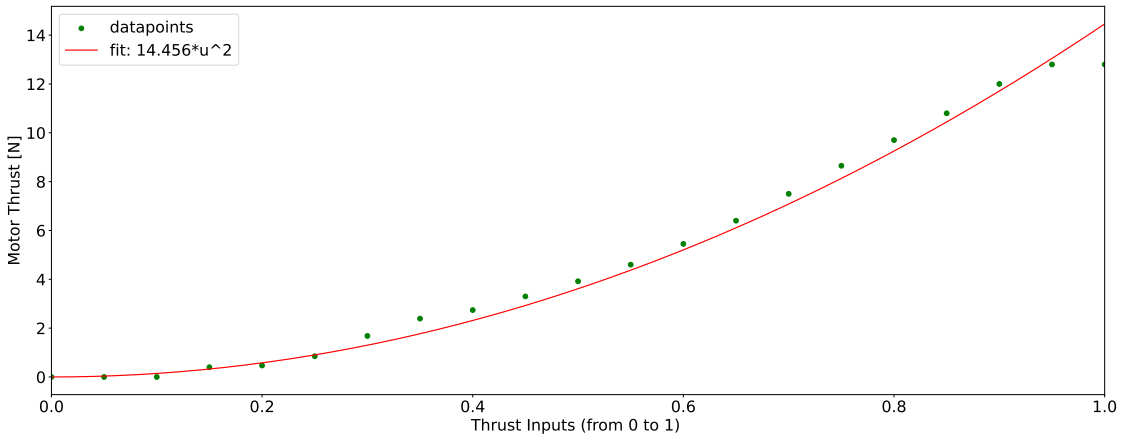

Figure 7: Experimental data points with a quadratic fit.

Overall Lift and Drag Finally, we need to determine the lift and drag caused by the airplane regardless of the inputs. We model the plane as an airfoil. According to wing theory by Durand (1935), an airfoil will cause drag and lift according to the relation seen in Equation (10). The lift and drag coefficients are dependent on the angle of attack of the plane. The angle of attack is defined as the difference between the pitch angel θ and the direction the plane is moving. The angle of the planes movement can be determined from the velocity vector. The aerodynamic coefficients need to be determined during flight experiments. By analyzing recorded flight data, an expression for $C_L(\alpha)$ and $C_F(\alpha)$ can be found. A_{tot} denotes the area of the airfoil. If it is not determined accurately, the experimental lift and drag coefficients can compensate for it. Fig. 8 shows the fitted lift and drag coefficients from experimental data out of a simulation. To get the plane to fly at different angles of attack, the speed has to be changed. If the plane flies fast, the angle of attack is low. In the contrary, if the plane flies slow, the angle of attack is high. To get meaningful data for the lift and drag coefficients, I flew the plane with different speeds. The different speeds are visible as the small clusters in Fig. 8. The simulation is described in more detail in Section 3.6.

$$\begin{aligned} F_{Lift} &= \frac{1}{2}\rho v^2 A_{tot} C_L(\alpha) \\ F_{Drag} &= \frac{1}{2}\rho v^2 A_{tot} C_D(\alpha) \end{aligned} \tag{10}$$

All the forces can be combined into one force and one torque acting on the Center of Mass of the plane. The torques are found by taking the cross product between the forces and the vector from the center of mass to the point of application of the force. A summary of all the forces can be seen in Equation (11), where the variables d_i denote the lever arm for the respective forces.

$$\begin{aligned} F_x &= 14.456 \cdot u_1^2 - \frac{1}{2}\rho v^2 A_{tot} C_D(\alpha) - g\sin(\theta) \\ F_y &= A_5 \rho u_5^2 v^2 + g\sin(\phi)\cos(\theta) \\ F_z &= \rho v^2 (A_2 u_2^2 + A_3 u_3^2 + A_4 u_4^2) - g\cos(\phi)\cos(\theta) + \frac{1}{2}\rho v^2 A_{tot} C_L(\alpha) \\ M_x &= \rho v^2 (A_2 u_2^2 d_2 - A_3 u_3^2 d_3) \\ M_y &= \rho v^2 (A_4 u_4^2 d_4 - A_2 u_2^2 d_2 - A_3 u_3^2 d_3) \\ M_z &= \rho v^2 A_5 u_5^2 d_5 \end{aligned} \tag{11}$$

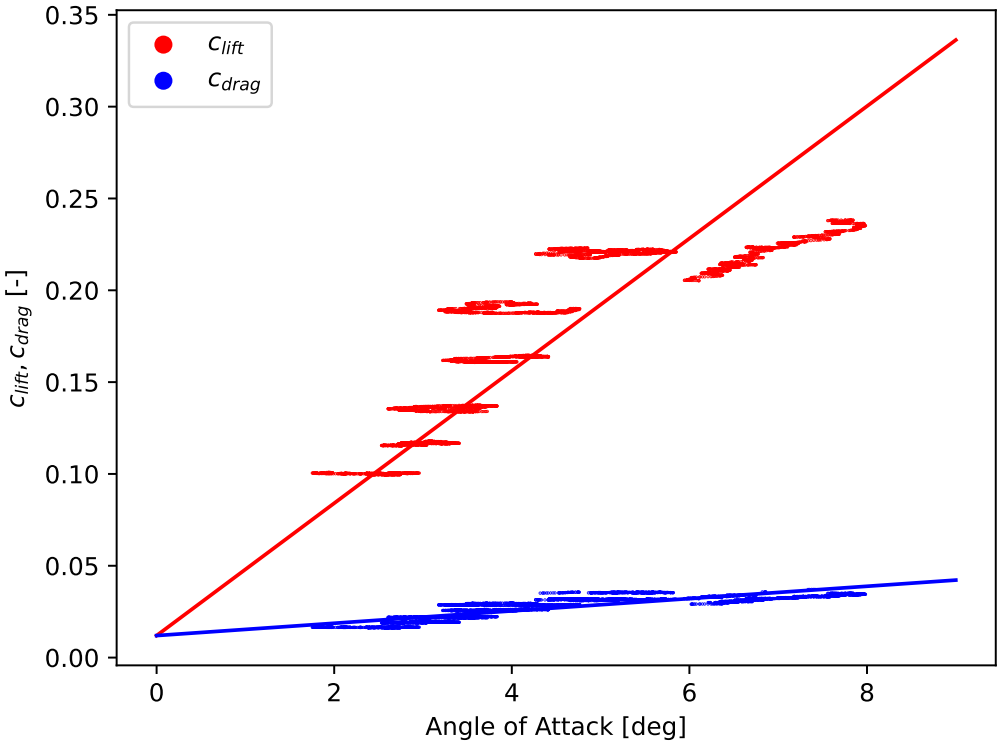

Figure 8: Experimental data points of a simulation (see Section 3.6) with a fit for both the lift and the drag coefficient.

3.3.5 Control Approach

We want to design a controller that can fly the desired trajectory as described in Section 3.2 and can perform the agile grasping maneuver. To achieve this, I propose splitting the control task into two different sequences. During the normal flight, we use the standard controller that is already implemented on the Pixhawk. When the object comes closer, the control mode is switched to the 2-Dimensional MPC controller that does a coordinated grasp-flight maneuver to achieve an agile eagle-like pickup. The MPC controller is discussed in Section 3.3.6. Fig. 9 shows how the controller should switch modes.

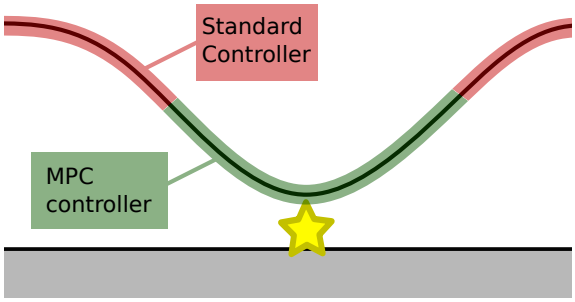

Figure 9: The controller switches modes according to the task that it has to accomplish.

3.3.6 Model Predictive Plane Controller

Due to the translational speeds required for flying, the pick-up of the object will be quite challenging. To coordinate the flight behavior with the gripper control, both the plane and the gripper should be actuated by the same controller. To make explanations easier, we assume that the pickup happens in the xz-plane and

the y coordinate stays constant. With a change in variables, this formulation can be used for a pick-up along an arbitrary direction. The model predictive control approach is discussed in the following steps:

2D Reduction We want to achieve the task most simply and reliably, therefore the MPC problem is reduced to a 2D problem in the pick-up plane. We use a PID controller to keep the aircraft in the pick-up plane, by controlling the yaw and roll to 0 and the y-coordinate to the object's y-coordinate. By only looking at the 2D problem, the simplifications in Equation (12) are made.

$$\begin{aligned} \phi = \psi = \dot{\psi} = \dot{\phi} = 0 \\ y = y_{obj} \\ u_4 >> u_2, u_3 \rightarrow u_2 = u_3 = 0 \end{aligned} \tag{12}$$

PID-Loop for Attitude Control The control problem can be simplified even further by combining a PID loop with the MPC. In the case of the airplane, this is a high-frequency PID loop for attitude control and an MPC for trajectory control and grasp coordination. The latter one runs at a lower frequency due to the numerical optimization problem, that has to be solved.

State Space and Inputs After reducing the problem to the 2D case, we end up with the state space variables and inputs seen in Table 3.

State Space Variable	Description	Input Variable	Description
x	x-Coordinate	u_{thrust}	Motor Thrust
z	z-Coordinate	u_{pitch}	Pitch Angle
θ	Plane Pitch	u_{arm}	Gripper Arm Torque
β	Gripper Arm Angle (relative to plane)	u_{grip}	Gripper Open/Close

Table 3: state space variables and inputs of the MPC-Problem.

Constraints The following constraints have to be satisfied by the optimizer:

- **C1** $v_{plane} > v_{stall}$ The plane needs to keep a certain speed to ensure that it has enough lift. The value of v_{stall} can be determined, by equating the lift with gravity.
- **C2** $\mathbf{u}(t) \in \mathbb{U}$ The control inputs can only be as big as the physical system allow them to be. This means that servos and motors are limited in their range which needs to be taken into account.
- **C3** $\mathbf{x}(t) \in \mathbb{X}$ The state space variables need to stay in a physically allowed range. For example, the gripper angle cannot take a value where the gripper would penetrate the aircraft. Also, the z-coordinate should not be negative and the pitch should stay in a reasonable range.
- **C4** $\mathbf{x}(t_0)$, $\mathbf{x}(t_f)$ Final and initial conditions have to be chosen, such that the standard controller can continue the flight after the pickup.

System Dynamics The whole system dynamics have been discussed in the sections 3.3.2 and 3.3.4. The reduced system dynamics are summed up in Equation (13). These equations will be used to predict the system's behavior in the model predictive controller. $\mathbf{y}$ denotes a set of convenient variables that are useful to penalize in the cost function. It contains the distance between the aircraft and the pick-up spot in the z-dimension and the speed of the gripper. Both are multiplied by the factor $(d_{pickup} - |x - x_{obj}|)$ which makes the value bigger, when the x-distance between the plane and the object decreases. The variable d_{pickup} denotes the length of the pick-up phase. With this factor, I hope to maximize the influence on the cost function of these values at the exact moment of pick-up. The angle of attack α

is defined as the difference between the pitch angle θ and the angle of the velocity vector.

$$\frac{d^2}{dt^2}\mathbf{x} = \frac{d^2}{dt^2}\begin{bmatrix} x \\ z \\ \theta \\ \beta \end{bmatrix} = \begin{bmatrix} -\frac{1}{2\cdot m}\rho v^2 A_{tot} C_D(\alpha) - \frac{g}{m}\sin(\theta) \\ \frac{1}{2\cdot m}\rho v^2 A_{tot} C_L(\alpha) - \frac{g}{m}\cos(\theta) \\ 0 \\ \frac{\cos(\theta+\beta)\cdot g}{m_{gripper}\cdot l_{arm}} \end{bmatrix} + \begin{bmatrix} \frac{14.456\cdot u_1^2}{m} \\ \frac{\rho v^2 A_4 u_4^2}{m} \\ \frac{\rho v^2 A_4 u_4^2 d_4}{I_{pitch}} \\ \frac{u_{arm}}{m_{gripper} l_{arm}^2} \end{bmatrix} \tag{13}$$

$$\mathbf{y} = \begin{bmatrix} y_z \\ y_{grip} \end{bmatrix} = \begin{bmatrix} \overbrace{(z_{gripper} - z_{obj})}^{\Delta z}\cdot(d_{pickup} - |x - x_{obj}|) \\ \underbrace{|\dot{\vec{x}} + \dot{\beta} \times \vec{r}_{plane-grip}|}_{\text{gripper speed}} \cdot(d_{pickup} - |x - x_{obj}|) \end{bmatrix}$$

Nonlinear Quadratic Optimal Control The nonlinear system dynamics are discretized and brought into a form seen in Equation (14).

$$\begin{aligned} \mathbf{x}_{n+1} &= f_d(\mathbf{x}_n, \mathbf{u}_n) \\ \mathbf{y}_n &= g_d(\mathbf{x}_n) \end{aligned} \tag{14}$$

With that, we can formulate the quadratic cost function seen in Equation (15), where $\mathbf{Q}$ and $\mathbf{R}$ are weight matrices for the different variables. Iteratively, the optimal weights are found, which should result in the desired pick-up discussed in Section 3.1. I expect, that a penalization of the gripper speed will cause the plane to slow down close to v_{stall} and swing the arm in the contrary direction of flight. The trajectory found by the optimizer is discussed in Section 4.2.

$$\mathbf{J}(\mathbf{x}_k, \mathbf{u}_k) = \sum_{i=0}^{N} (\mathbf{y}_{k+i}^\top \mathbf{Q} \mathbf{y}_{k+i} + \mathbf{u}_{k+i}^\top \mathbf{R} \mathbf{u}_{k+i}) \tag{15}$$

As seen in Equation (16), the optimal control inputs are found by optimizing over the cost function $\mathbf{J}$.

$$\mathbf{U}_k^* \left(x_k\right) := \arg\min_{\mathbf{U}_k} \mathbf{J}\left(\mathbf{x}_k, \mathbf{U}_k\right) \tag{16}$$

3.4 Controller Architecture

The MPC runs on the Raspberry Pi and gives control inputs to the plane over the Pixhawk. It can also actuate the gripper through the Arduino. An extended Kalman filter (EKF) is used for state estimation. It runs on the Pixhawk because the firmware has great ready-to-use implementation. To function properly the EKF needs sensor information from both the real-time kinematic positioning system (RTK GPS) as well as the inertial measurement unit (IMU). A schematic view of the control architecture can be seen in Fig. 10. The Arduino is needed to actuate the gripper because the Raspberry Pi is not suited to directly actuate the motors and servos in the gripper. To make the control problem easier, it makes sense to use PID loops running at a high frequency for attitude control. We can also use PID control loops to ensure that the airplane stays in the pick-up plane and does not deviate in the y-coordinate. Finally, this leaves us with the throttle, pitch, and arm commands to control the grasp procedure.

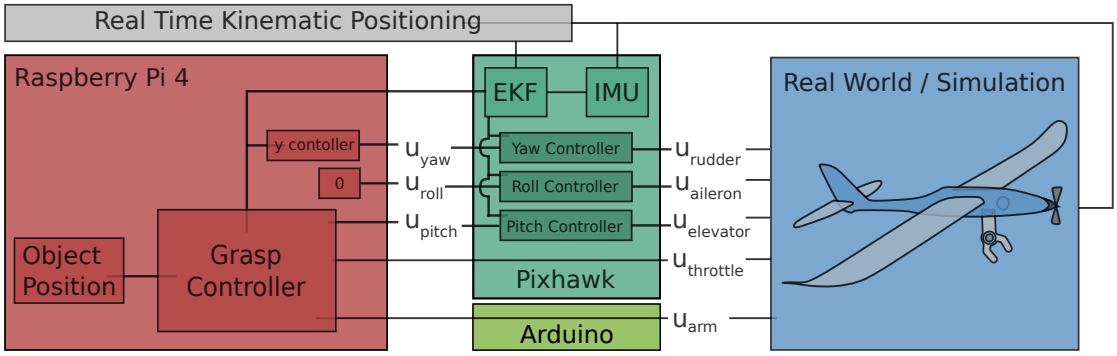

Figure 10: The proposed control architecture with a grasp controller running on the companion computer and an EKF running on the Pixhawk. The components on the Pixhawk are already implemented, whereas the elements on the Raspberry are newly developed.

3.5 System Architecture

The system architecture is inspired by the work of Appius et al. (2022), who used a similar approach for the dynamic grasping of objects with quadcopters. The following list should clarify the components' functions and interconnections. A schematic view of the system architecture can be seen in Fig. 11.

Real Time Kinematic Positioning (RTK GPS) is used for determining the position of the aircraft. The system works by comparing the GPS signal of a fixed ground station with the GPS signal on the aircraft. With this, centimeter-level accuracy can be achieved at a frequency of 20 Hertz. This is enough to accurately control the aircraft's position in a pick-up task. It works out-of-the-box with the Pixhawk flight controller.

Arduino The Arduino is used to actuate the gripper. It communicates with the Raspberry Pi over a serial connection. The Arduino is needed because it is impractical and inaccurate to directly actuate motors from the Raspberry's GPIO pins.

Raspberry Pi The Raspberry Pi is used as a companion computer for the flight controller. It runs the MPC Controller and sends motor commands to the flight controller. It reads state information from the Flight controller's EKF.

Pixhawk The Pixhawk is the flight controller. It is used to make the actuation of the motors easier and to get state estimation from the built-in EKF, which uses information from the IMU and RTK GPS. The raspberry and the Pixhawk communicate over a serial connection.

Ground Computer The ground computer is used as an interface for the user. High-level commands are sent to the Raspberry Pi over an SSH session. The ground computer and Raspberry Pi communicate over 433 MHz SiK radio.

RC Remote The radio control remote is used as a secondary communication channel, which can be used for testing or emergencies. It does not communicate over the raspberry pi but directly with the Pixhawk flight controller.

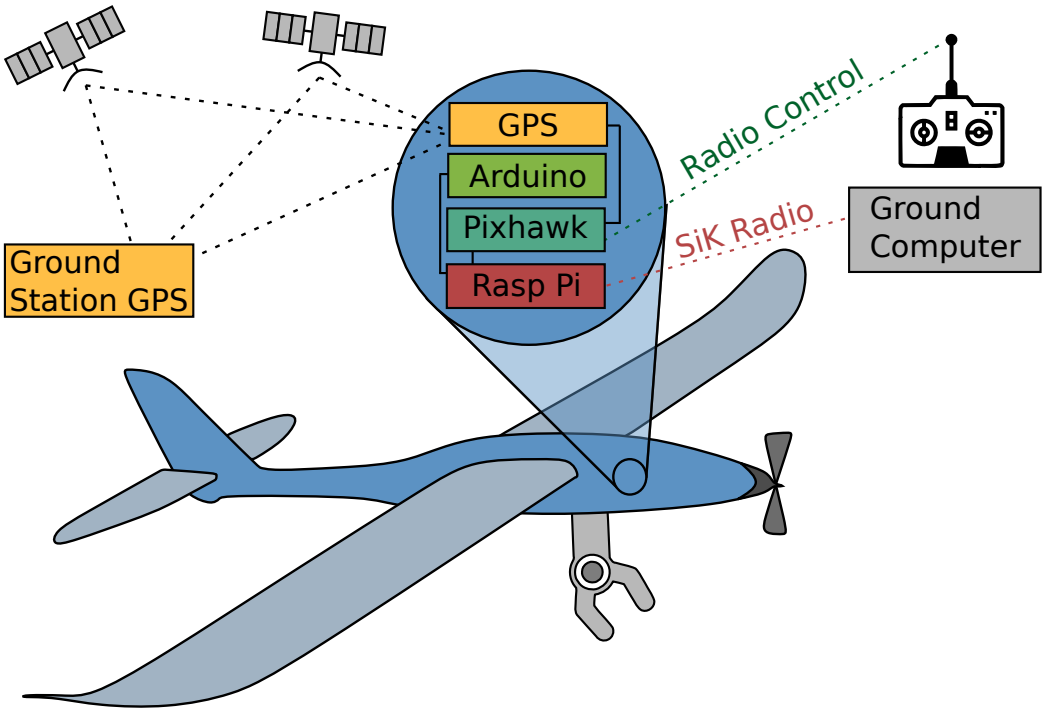

Figure 11: A schematic view of the system architecture with all its components and connections

3.6 Gazebo Simulation

The development of a new controller requires a lot of testing. In order to perform a lot of tests without risking destroying the system itself, simulation software is needed. It allows extensive testing of new software before deploying it on the real system, which can save a lot of time and resources. Luckily, there are many existing open-source simulators which I could reuse for this project. The PX4 community, which develops flight controllers also develops simulators to test these controllers. These simulators simulate both the physics as well as the PX4-based controllers, which is called System-In-The-Loop (SITL). The simulation is based on Gazebo, which is a popular open-source 3D robotics simulator. It uses different plugins to implement the flight controller and aerodynamic effects. Fig. 12 shows an overview of the simulation architecture. When comparing it to the real architecture seen in Fig. 10, we see that most of the features are exactly the same. I had to customize the simulator to also include the gripper's physics. To achieve that, I wrote a custom ROS2-based plugin that allowed me to communicate with the simulator over a separate channel. This allowed me to send arm and gripper commands to the simulator while simultaneously sending pitch, roll, and yaw commands over MAVSDK. I needed this second link because the MAVSDK framework for sending the flight commands to the simulator does not easily allow adding new custom messages. Much effort was needed to analyze and understand the existing simulation framework and add new elements to it. I could not implement a working collision detection in the gripper simulation. The gazebo collision detection engine does not work very well when two objects collide at high speeds like they do in a fixed-wing pick-up. I defined a proximity-based measure to classify a grasp as successful which will be discussed in Section 6.3.

3.6.1 camera plugin

In order to get a nice view of the grasping process, I had to place a virtual camera beside the object. Unfortunately, the Gazebo version that was used in the PX4 simulator did not support adding custom cameras so I had to write a plug-in for that as well. I achieved this by analyzing the source code of an Intel Real-Sense gazebo plugin. I could pick out the elements I needed for my camera and could modify them according to my needs. Finally, the new plugin allowed me to export high-quality videos of my simulated grasps. A still image sequence of such an exported video can be seen in Fig. 23.

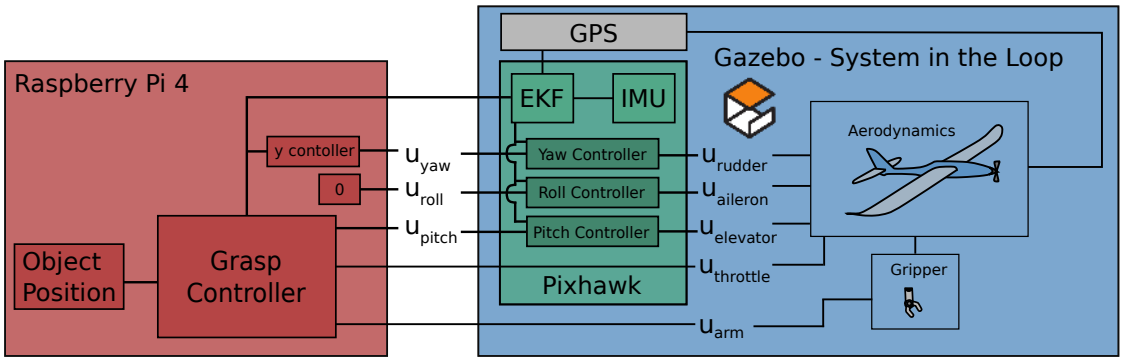

Figure 12: A schematic of the gazebo-based simulator architecture.

4 Controller Implementations

To achieve the goal of an autonomous fixed-wing pick-up, I wanted to use a model predictive controller. I started working and implementing based on the concepts described in Section 3. For the reasons mentioned in Section 4.1, the MPC approach did not yield satisfactory results so I had to try a new approach, which is described in Section 4.2. The following two subsections give an insight into the development process and how I resolved the problems of the MPC approach with a new control method.

4.1 MPC Implementation

In order to get a working MPC, an optimization problem has to be posed and solved. As described in the methodology section, this involves minimizing a cost function over a finite time horizon. To pose the problem and numerically find a minimum, I used the CasADI framework developed by Andersson et al. (2019). It is a general-purpose tool that can be used for non-linear optimization.
Once I could solve the optimization problem, I had to combine it with a second piece of code that would use the generated optimal inputs and feed them to the PX4 controller. Because the optimization process took more time than a single control iteration, I had to write a framework that would feed control inputs and simultaneously optimize.
To do this continuously with a predefined frequency, I had to use a parallel programming approach. I would run the optimizer in one thread and the control-input forwarder in another. Once the optimization process is finished, it would notify the forwarder that new optimal control strategies are available. After that, the optimizer would start again with new initial conditions. This procedure would continue until the grasping process was finished. Fig. 13 shows the explained procedure graphically.

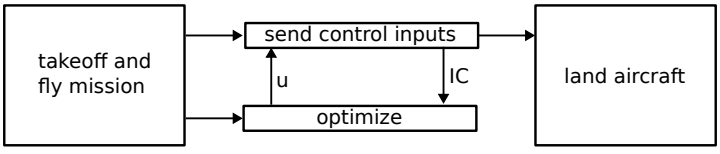

Figure 13: This illustration shows the use of multi-threading in the MPC implementation. After a successful takeoff and approach, the MPC-grasper is deployed. It runs the optimizer and input forwarded simultaneously. After the grasp is completed, the landing sequence is initiated.

Unfortunately, this approach did not yield satisfactory results. The optimization process took too long to converge. I tried to make it faster by making simplifications in the model and tuning the time horizon and other parameters. For example, I modeled the aerodynamic drag and lift as a constant and not as a function dependent on the angle of attack. This resulted in faster convergence time, but also in a less accurate model. I invested a lot of time in trying to find a sweet spot between accuracy and speed. But unfortunately, this sweet spot was not reachable with my hardware and model. Table 4 illustrates the inversely proportional

relation between accuracy and convergence speed.

Model	Accuracy	Convergence Speed
Complex Model	GREAT	SLOW
Simplified Model	BAD	VERY FAST
Sweet Spot Model	GOOD	FAST

Table 4: The problematic properties of the MPC approach is described in a tabular fashion. The accuracy of the model and its convergence speed are inversely proportional, which makes it hard or even impossible to find a sweet spot model.

4.2 Offline Trajectory Generation

The convergence speed problem of the MPC approach made me think about other ways to solve the problem. The optimizer came up with good solutions, they just came too slow to use them in a real-time application. An example of an optimal trajectory calculated by the optimizer can be seen in Fig. 14. It is visible that the plane optimizes its flight such that the gripper gets very close to the object. The arm-swinging motion also slows down the gripper as much as possible. It does that in a very similar way to the eagle.

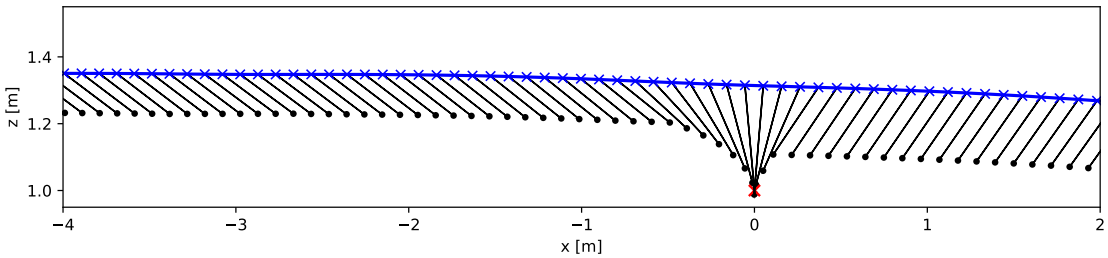

Figure 14: The optimal trajectory of the plane is visualized as a line. The gripper and arm can be seen in black.

The optimal trajectory is calculated based on some initial conditions, where the grasp sequence started. In the case of Fig. 14 it is a height of 1.35 m and a speed of 10 m/s. Because of disturbances, the plane will not always approach the object in the same way and the initial conditions for the grasp will be different. So instead of calculating the trajectories while flying, we could calculate it beforehand and just look it up when approaching the object. This is also called offline trajectory generation.
I discretized the initial heights and initial speeds for some reasonable ranges and calculated the optimal trajectories. Fig. 15 shows the optimal trajectory of the plane for different heights.

The whole grasp procedure is visualized in Fig. 16. As mentioned above, the optimal trajectories are computed before the flight and stored in a file. After a successful takeoff and approach, the grasping controller is deployed. It checks the current height and speed and loads the appropriate optimal trajectory from the file. Afterward, this trajectory is executed and the landing sequence gets initiated after the grasp. The trajectory is flown by feed-forwarding the optimal control inputs found by the optimizer. To account for disturbances and noise, we also include a feedback term that compares the current state with the optimal state and introduces a feedback term to the optimal control solution.

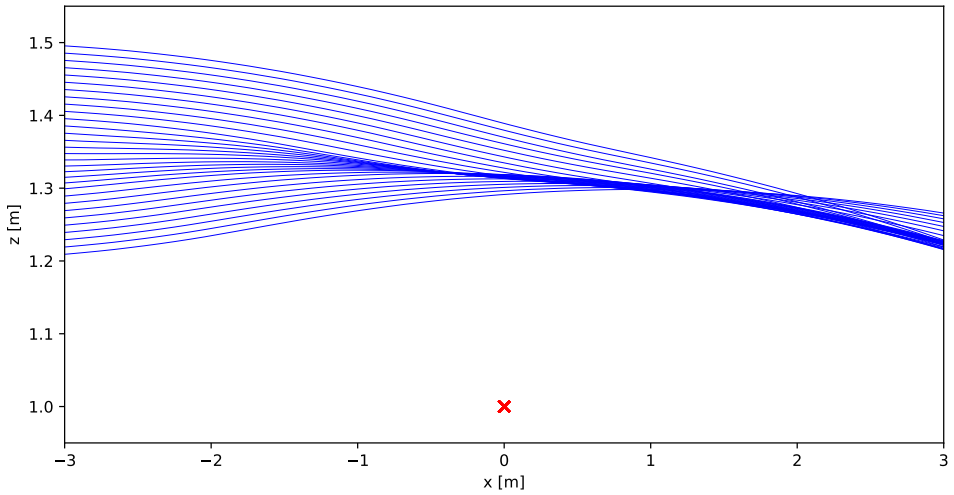

Figure 15: The optimal trajectory of the plane for the grasp for different initial heights.

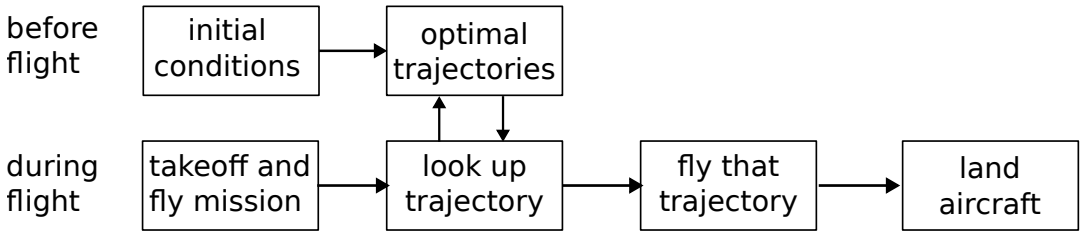

Figure 16: This illustration shows the concept of the offline trajectory controller.

5 Building the System

The system was built on the base of a Multiplex Funcub XL, which can be seen in figure Fig. 17. It was available as a building kit that could be assembled to a plane. This allowed me to fit all the components mentioned in Section 3.5 into the plane. It was challenging to put all the components efficiently into the small amount of space available in the aircraft.

Figure 17: A picture of the Multiplex Funcub XL model airplane (RC-Planes.nl, n.d.)

5.1 Gripper

The design, development of the gripper used for the FALCON plane were made by Marc Blöchlinger (mbloechli@ethz.ch). He studied the requirements and compared different gripper types for the grasping task. He then chose a FinRay based design, that would be attached to the front of the arm. Iteratively, he optimized the gripper and finally came up with the working prototype that can be seen in Section 6.2.

6 Experiments and Results

The flight control architecture was tested both in the real world as well as in simulation. Both scenarios are described in the following two sections.

6.1 Real World Flights

To perform the experiments, we could use a model plane airfield in Dietikon near Zürich. The airfield consisted of a big grass plane that was trimmed frequently to ensure an even surface. This allowed our plane to roll, take off and land there. We used the Pixhawk 4 controller to perform basic flight maneuvers autonomously. The takeoff as well as flying through predefined checkpoints worked as expected. The autonomous landing sequence did not work properly and still needs further investigation and tuning. I was not able to build the whole setup successfully in the scope of this semester project. The following figures 18, 19, and 20 show a successful takeoff and flyby as well as a landing attempt. The videos were filmed using an ordinary smartphone and required some zoom, which lead to low resolutions for some sequences.

Figure 18: A sequence of pictures from a video of a successful takeoff.

Figure 19: A sequence of pictures showing the plane flying by.

Figure 20: A sequence of pictures showing an unsuccessful landing attempt

6.1.1 Fixing the aircraft

During the setup of the plane's autonomous maneuvers seen in the figures in Section 6.1, the plane had to endure some crashes. Errors in the software configuration led to crashes during the development. Even though I worked very cautiously and tried to minimize hardware damage, testing proved to be risky and difficult. Wind conditions as well as small undetected malfunctions on the plane could result in a crash. An example of a particularly horrible crash can be seen in Fig. 21. Because of limited time and resources, I could not buy a new plane so I had to fix it. This required a lot of creativity, hardware skills, and patience. With a combination of hot glue and 3D-Printed spare parts, I could always fix the plane and make it airborne again.

Figure 21: A hardware malfunction in the left aileron made the plane uncontrollable and therefore leave its trajectory. This immediately made me kill the motors which made the plane glide into a tree and crash. This was a very severe crash that resulted in a break of the airplane's body. Luckily, it fell into the tree and not into the nearby river seen in the background.

6.2 Dry Runs

To test the performance of the gripper without flying the aircraft, I performed a dry run. Instead of flying, I took the aircraft manually and ran across the room. These tests were performed in a room equipped with a Vicon Motion Capture System. This allowed me to actuate the arm in a way that was similar to real flight

grasping. If the object and plane position is known, the arm angle can be calculated by using the relation seen in Equation (17). This angle is given to the servo controlling the arm.

$$\tan\theta = \frac{z_{plane} - z_{obj}}{x_{plane} - x_{obj}} \tag{17}$$

By running by the object with the plane, a pick-up is simulated. I could achieve a running speed of approximately $6m/s$. This is less than the planes' $9 - 10m/s$, but still a useful result. Because of the human factor involved in this experiment, it cannot be reproduced accurately and therefore no statement about the robustness of the gripper can be made in this context. The dry run grasp can be seen in Fig. 22.

Figure 22: A sequence of images showing a dry run experiment of the gripper and the plane.

6.3 Simulated Flights

Simulators allow faster and cheaper testing of new controllers. With the press of a button, a simulator can be restarted when things go wrong. In the real world, failures result in costly and time-intensive repairs. With that incentive in mind, I tested all my code first in simulation before going to the real world. The custom simulator used for this is described in Section 3.6. With a simulated camera, I could export a video of a grasping sequence viewed from the side. Some still images of that video can be seen in figure 23.

Figure 23: A sequence of images showing a successful grasp in simulation. As mentioned in Section 3.6, there is no collision detection implemented so the object just stays at the spot.

To assess the robustness of the approach, I conducted 10 experiments and logged the data. The gazebo simulator does also simulate sensor noise and GPS estimation errors. Due to these disturbances, the trajectories had different initial conditions for which the controller had to adjust. All the simulated tests used an object height of 10 meters. This height was chosen because it allowed a faster approach phase. Because the plane uses a minimum takeoff altitude of 10 meters, it is convenient to stay at that height and not adjust it before the grasp. This does not mean that lower pick-up altitudes are not possible. They just take a little longer because of the required height adjustment before the pick-up. Fig. 25 shows the 10 different runs as well as the reference trajectory for each run. Fig. 24 shows a zoomed version of the plot around the object, which allows a better assessment of the grasps' success.

#Grasp	z_0 [m]	$z_{0,traj}$ [m]	v_0 [m/s]	$v_{0,traj}$ [m/s]	Grasp Success
01	10.3744	10.37	9.94	10.0	YES
02	10.3549	10.35	9.88	10.0	YES
03	10.4008	10.4	9.9	10.0	YES
04	10.3733	10.37	9.89	10.0	YES
05	10.3232	10.32	9.92	10.0	YES
06	10.5414	10.5	9.88	10.0	NO
07	10.3673	10.37	9.91	10.0	YES
08	10.5011	10.5	9.88	10.0	NO
09	10.2325	10.23	9.93	10.0	NO
10	10.2574	10.26	9.9	10.0	NO

Table 5: Table showing initial conditions as well as chosen trajectory initial conditions for the robustness tests. The last column also shows if a grasp was successful according to the definition in Section 6.3.

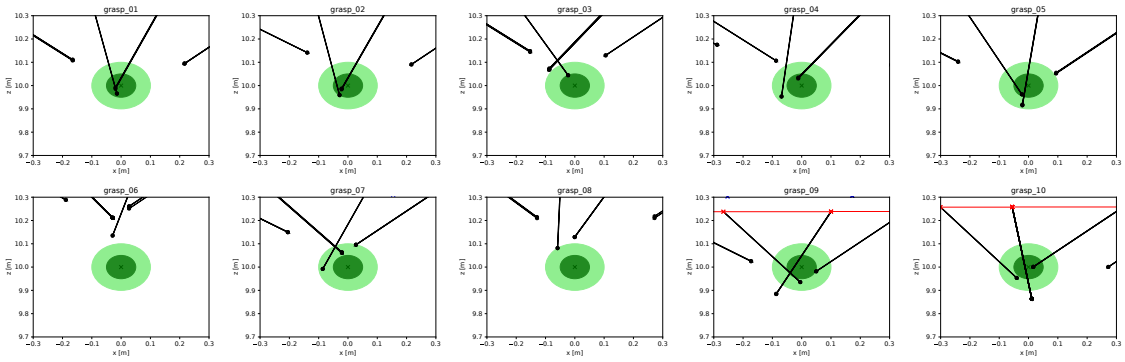

Figure 24: Close-up plots of the 10 runs conducted for robustness testing. The green ball shows a 5 cm and 10 cm radius around the object which is used to classify a successful grasp. AS in Fig. 25, the black sticks with dots represent the gripper and the arm.

Some metrics of the robustness test can be seen in Table 5. As described in Section 4.2, the controller will check the initial conditions at the start and pick the most appropriate precomputed trajectory. When looking at the values, we see that there is a low variance for the speeds over the different experiments. The trajectory for the initial velocity of 10 m/s is always picked. On the other hand, there is higher variance across the initial height and the controller chooses different optimal trajectories for almost all approaches.
Because the simulator does not simulate the collisions between the gripper and the object, the success of a grasp has to be determined differently. A very simple method is to look at the position of the gripper and its proximity to the object. In this test, I defined a grasp as successful if the gripper would come into 10 cm proximity of the object. This is a very generous way of defining a successful grasp. There are a lot more factors determining a grasp's success, rather than only the distance of the gripper to the object. These factors include orientation, impact speed, friction coefficients, and many more. But for this early-stage controller evaluation, the proximity-based evaluation method is fine.

When evaluating the different runs we find that **the controller achieves a success rate of 60 %** for grasping in simulation.

7 Discussion

7.1 Key Results

Through months of development, I could present a working platform that can be used for autonomous pick-ups. With the exception of the unreliable landing sequence, the aircraft can take off and fly on arbitrary trajectories. The chosen architecture also allows easy deployment of a custom controller.

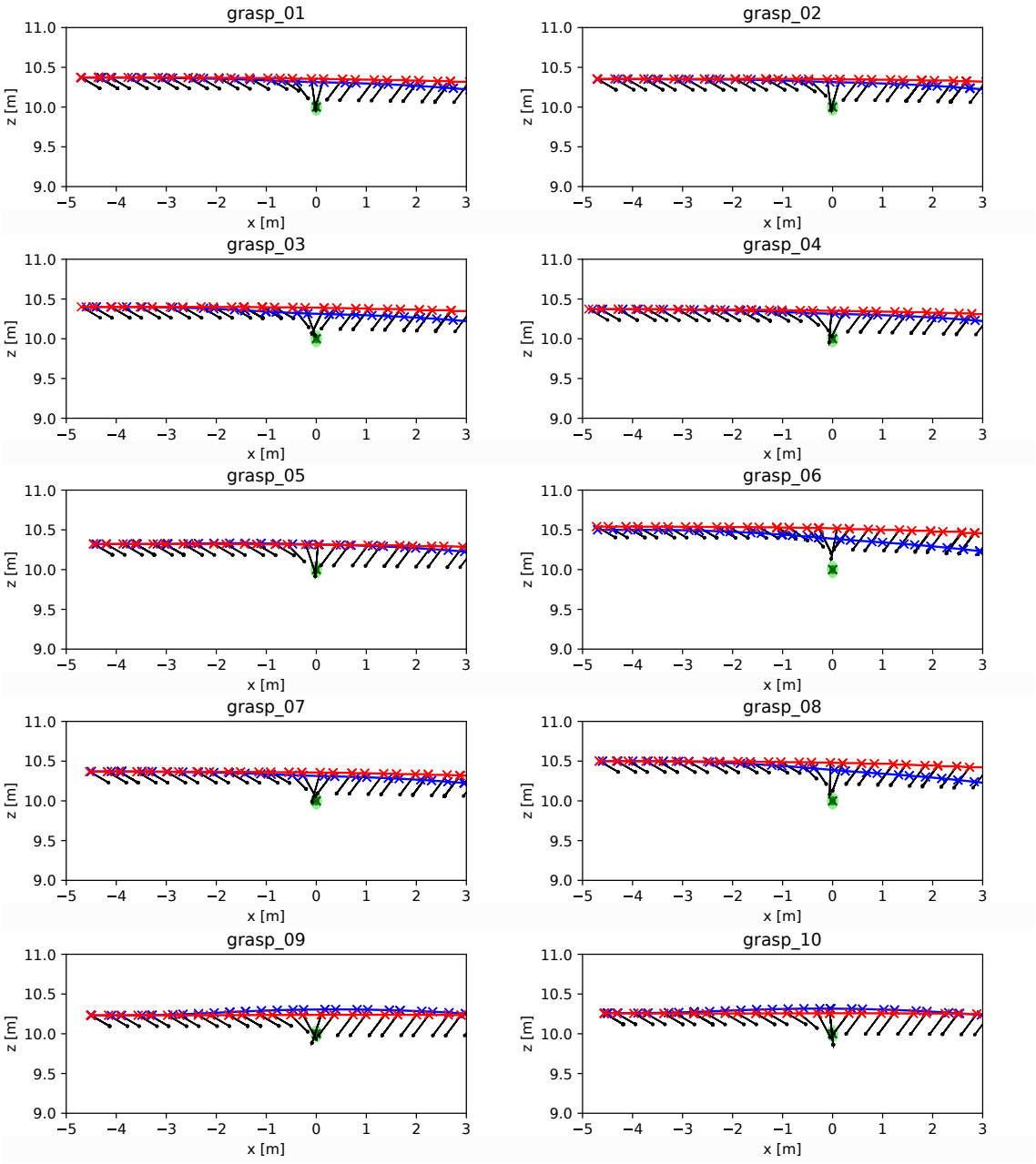

Figure 25: Plots of the 10 runs conducted as part of the robustness test. The object is marked as a green ball, the gripper position is indicated as a black stick with a dot at the front. The red path is measured data from the simulation and the blue path is the optimal trajectory.

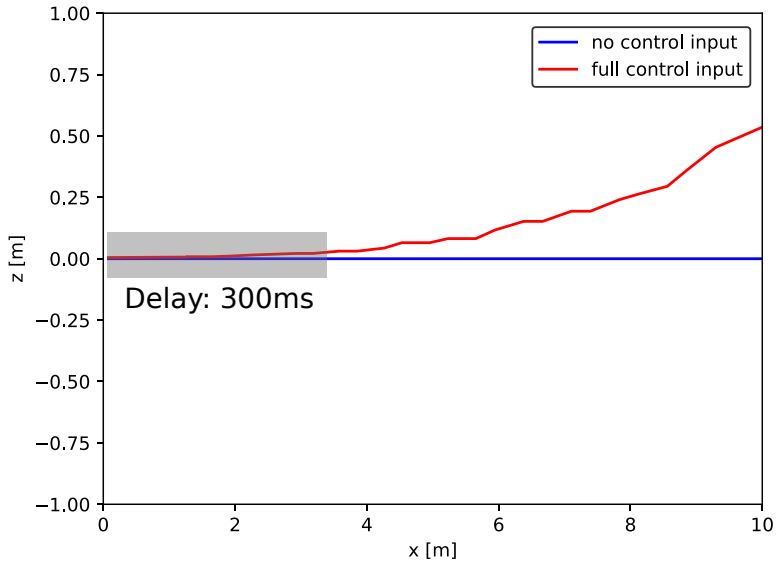

Figure 26: This plot demonstrates the slow dynamics in the height dimension.

Furthermore, I set up a custom simulation environment which allowed me to test and evaluate different pick-up controllers. I developed an optimizer to calculate optimal pick-up trajectories based on my mathematical analysis of the system. I used this optimizer to first try an MPC approach and then an offline trajectory approach. For the latter one, I could achieve a **60 %** pick-up success rate in simulation. With that, I almost reached my goal of 70 % which I set for myself at the begging of the project.
Additionally, I also tested the gripper on the real plane in a dry run to verify the functionality of Marc's gripper.
Although my research did not result in a successful fixed-wing pickup, I worked a substantial part in the direction of making it possible.

7.2 Controllability Issues

During the development and after looking at the first preliminary results, I could identify the biggest challenge for fixed-wing pickup. It is the very limited controllability of the plane. I tried to visualize this behavior in Fig. 26, by comparing a full input of the elevators with no input at all. As expected, the plane starts to rise when the elevators are actuated, but it does this only after approximately 300 ms. This may not seem like a lot, but because the plane travels at 10 m/s, this delay means that the plane only moves upwards 3 meters later. This is a big issue because the controller can't adjust for disturbances in the z-axis during that period. If we recognize an altitude error, the controller will counteract, but the result will only be noticeable 3 meters later. In my opinion, this issue makes the fixed-wing grasp almost impossible. A proposed solution for that problem would be a more agile gripper arm, that can also compensate for vertical errors. This would allow a fast error correction during the approach phase and a higher grasp success rate.

7.3 Grasp Definition

The definition of a successful grasp in this thesis is quite vague. By just looking at the distance between the gripper head and the object, I was disregarding a lot of other factors that play a role in a successful grasp. The orientation of the gripper during pick-up plays an important role. If the object does not go in between the two fingers of the gripper, it will be pushed away and not grasped. Another factor is the friction coefficient of the gripper.
The best way to measure gripper success in a simulation would be working collision detection. I tried to implement different collision engines in the simulation, but the high speeds of the pickup were not suitable and did not yield satisfying results. To simulate the behavior of the soft fin-ray gripper, a soft material model would have to be included in the simulation.

7.4 A very hard Challenge

I want to emphasize at this point, that the challenge of fixed-wing aerial grasping is very hard. When I proposed the project, the workload seemed manageable. After starting to work on some of the first challenges, I realized that the complexity was very high and demanding. Designing and building a system and then implementing a control approach is a very resource and time-intensive process. I worked a lot throughout the whole semester and could achieve some milestones, but it is still a long way to go. I also feel like my resources are quite exhausted after having done this first part of the project. I am lacking both experience and supervision from an expert in the field to complete such an ambitious project. I have invested a lot of time in the project and tried my best in making this work, but I feel like it is too hard for me to complete. Still, I am hoping for other people to continue where I left off and make it work one day.

7.5 Further Work (Near Future)

To make a first prototype of fixed-wing aerial pickup work, the following steps would have to be addressed:

Landing Issue The plane is not able to land reliably in its current form. Further configuration and setup are needed to make it work. Another option would be the use of an expert to take over the landing maneuver.

Takeoff with Gripper I did not try a real-world takeoff with the gripper attached to the plane. The distance between the gripper and the ground during takeoff is very low. This could be very problematic, in case of a slightly uneven takeoff surface.

Controllability Issue As explained in Section 7.2, the height of the plane is only adapting very slowly to the control inputs. This makes disturbance rejection very ineffective and a grasp unlikely. One way to counteract this would be to just abandon unsuccessful attempts and try again. Another method would be to adapt the hardware to allow more errors in the height of the controller.

Control Robustness The controller is in an early stage of development and not robust for every situation that could come up during a grasp experiment. Better mathematical models that would cover more effects, could make the plane more precise and a successful pickup likelier.

Unforeseeable Issues Of course, there are always issues that come up during later development that can't be seen in the beginning. I anticipate that a lot of them will show up when trying to perform the sim to real transfer for the controller.

7.6 Further Work (Far Future)

A successful proof of concept of fixed-wing aerial grasping would have many possible further developments that could make successful business cases. Aerial package delivery over longer distances, probe extractions at remote locations like the arctic, or removal of trash could all be applications. As already mentioned, the power efficiency of fixed-wing aircraft is far superior when compared with quadcopters which would enable more autonomy and longer operating times.
From a research standpoint, showing that fixed-wing aerial manipulation is possible could spark more researchers' interest in putting their manipulators on fixed-wing aircraft.
Future work would be to not only grasp objects that lie still on the ground but also moving objects or even flying ones. The control framework proposed in this thesis would still be useful, but some changes in the trajectory generation would probably be necessary. Another further improvement would be to capture objects, without placing them in a predefined position. To achieve this, a perception algorithm to detect the object would be needed. A way for the system to dodge objects like trees or houses would also be an important future work package. This could be done by using a self-localization and mapping package (SLAM) which could use event cameras.
In this thesis, we neglected the influence of winds. The robot will only pick up objects in wind-still conditions. In case of stronger winds, the controller would need to be updated to compensate for side, back, and front winds during both flight and pickup. By attaching airspeed sensors to the plane, the difference between the ground speed determined by GPS and the airspeed can be used to estimate wind direction and strength.
To go even further towards nature, the plane architecture could be replaced with a bird-like ornithopter. This is an aircraft that flies by flapping its wings. A grasp could probably be made even more energy efficient and

faster when working with such type of aircraft. An example of such an aircraft was proposed by Folkertsma et al. (2017).
All in all, there is still a lot of work to do when it comes to bio-inspired grasping. Real-world fully alive eagles are still miles ahead when compared to the most modern grasping robots.

8 Conclusion

This thesis summarized the state of the art for fixed-wing aerial grasping and proposed a promising system design and control approach to solve it. I designed and built a system that could achieve fixed-wing grasping. I analyzed the dynamics and wrote an optimization algorithm that generates optimal trajectories for grasping. Afterward, I deployed both an MPC as well as an offline trajectory controller in a customized gazebo simulation. The offline trajectory controller could reach a 60 % grasping rate, while the MPC was not fast enough. Furthermore, various flight tests were conducted on a nearby airfield to configure the system.
This work provides a basic direction and preliminary results for future researchers trying to achieve fixed-wing grasping.

References

Andersson, J. A. E., Gillis, J., Horn, G., Rawlings, J. B., & Diehl, M. (2019). CasADi – A software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, *11*(1), 1–36. https://doi.org/10.1007/s12532-018-0139-4 (cit. on p. 14)

Appius, A. X., Bauer, E., Blochlinger, M., Kalra, A., Oberson, R., Raayatsanati, A., Strauch, P., Suresh, S., von Salis, M., & Katzschmann, R. K. (2022). Raptor: Rapid aerial pickup and transport of objects by robots. *ArXiv*, *abs/2203.03018* (cit. on pp. 1, 12).

Durand, W. F. (1935). Basic ideas of wing theory: Flow around an airfoil. In W. F. Durand (Ed.), *Aerodynamic theory: A general review of progress under a grant of the guggenheim fund for the promotion of aeronautics* (pp. 1–24). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-91485-0_1. (Cit. on p. 8)

Fiaz, U. A., Abdelkader, M., & Shamma, J. S. (2018). An intelligent gripper design for autonomous aerial transport with passive magnetic grasping and dual-impulsive release. *2018 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, 1027–1032 (cit. on p. 1).

Fishman, J., Ubellacker, S., Hughes, N., & Carlone, L. (2021). Dynamic grasping with a "soft" drone: From theory to practice. *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 4214–4221 (cit. on p. 1).

Folkertsma, G. A., Straatman, W., Nijenhuis, N., Venner, C. H., & Stramigioli, S. (2017). Robird: A robotic bird of prey. *IEEE Robotics & Automation Magazine*, *24*, 22–29 (cit. on pp. 2, 24).

Gawel, A., Kamel, M., Novkovic, T., Widauer, J., Schindler, D., von Altishofen, B. P., Siegwart, R. Y., & Nieto, J. I. (2017). Aerial picking and delivery of magnetic objects with mavs. *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 5746–5752 (cit. on p. 1).

Hingston, L., Mace, J., Buzzatto, J., & Liarokapis, M. (2020). Reconfigurable, adaptive, lightweight grasping mechanisms for aerial robotic platforms. *2020 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, 169–175 (cit. on p. 1).

Karydis, K., & Kumar, V. R. (2017). Energetics in robotic flight at small scales. *Interface Focus*, *7* (cit. on p. 1).

Lab, G. (2013). Avian-inspired grasping for quadrotor micro uavs (cit. on p. 2).

McLaren, A. I., Fitzgerald, Z., Gao, G., & Liarokapis, M. (2019). A passive closing, tendon driven, adaptive robot hand for ultra-fast, aerial grasping and perching. *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 5602–5607 (cit. on p. 1).

Meng, J., Buzzatto, J., Liu, Y., & Liarokapis, M. (2021). On aerial robots with grasping and perching capabilities: A comprehensive review. *Frontiers in Robotics and AI*, *8* (cit. on p. 2).

New York Post. (2017). *Bald eagle snatches salmon right off fisherman's boat*. Retrieved October 3, 2022, from https://nypost.com/video/bald-eagle-snatches-fishermans-salmon-right-off-his-boat/. (Cit. on p. 1)

Noth, A., Bouabdallah, S., & Siegwart, R. Y. (2006). Dynamic modeling of fixed-wing uavs (cit. on p. 4).

RC-Planes.nl. (n.d.). *Multiplex rr funcub xl nd*. Retrieved January 16, 2023, from https://www.rc-planes.nl/fr/product/multiplex-rr-funcub-xl-nd/. (Cit. on p. 17)

Roderick, W. R. T., Cutkosky, M. R., & Lentink, D. (2021). Bird-inspired dynamic grasping and perching in arboreal environments. *Science Robotics*, *6* (cit. on pp. 1, 2).

Stewart, W., Ajanic, E., Muller, M., & Floreano, D. (2022). How to swoop and grasp like a bird with a passive claw for a high-speed grasping. *IEEE/ASME Transactions on Mechatronics* (cit. on pp. 1, 2).

Thomas, J. R., Loianno, G., Polin, J., Sreenath, K., & Kumar, V. R. (2014). Toward autonomous avian-inspired grasping for micro aerial vehicles. *Bioinspiration & Biomimetics*, *9* (cit. on p. 1).

Zhang, H., Sun, J., & Zhao, J. (2019). Compliant bistable gripper for aerial perching and grasping. *2019 International Conference on Robotics and Automation (ICRA)*, 1248–1253 (cit. on p. 1).

Zhu, Y., Guo, Q., Tang, Y., Zhu, X., He, Y., Huang, H., & Luo, S. (2022). Cfd simulation and measurement of the downwash airflow of a quadrotor plant protection uav during operation. *Computers and Electronics in Agriculture* (cit. on p. 1).